UNITED STATES PATENT APPLICATION

for

DETERMINISTIC HARDWARE RESET FOR FRC MACHINE

Inventors:

Steven J. Tu
3527 East Windmere Drive
Phoenix, AZ 85048
Citizen of The United States

Hang T. Nguyen
8613 S. Dorsey Lane
Tempe, AZ 85284
Citizen of The United States

File No.: 42390.P12490

"Express Mail" mailing label number: EL 867 651 416 US
Date of Deposit: **December 31, 2001**
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, Washington, D. C. 20231

Maureen R. Pettibone
(Typed or printed name of person mailing paper or fee)

Maureen R. Pettibone
(Signature of person mailing paper or fee)

12/31/01
(Date signed)

DETERMINISTIC HARDWARE RESET FOR FRC MACHINE

Background of the Invention

[0001] Technical Field The present invention relates to microprocessors and, in particular, to mechanisms for resetting the machine states of processors.

[0002] Background Art. Reset refers to the mechanism by which a computer system is put into a reference state. A computer system may be reset when it is first turned on, or an operating computer system may be reset in response to events, such as the occurrence of certain error conditions. At reset, the central processing unit (CPU or processor) of the computer system is typically put into a particular machine state. For a computer system that does not support functional redundancy checking (FRC), this state may be defined "loosely". For example, a reset pin may force a relatively small fraction of the processor nodes, e.g. latches, flip flops and the like, to specified logic states, with the states of the remaining nodes left indeterminate. For such non-FRC systems, the indeterminate states of the unspecified nodes have no impact on the visible results produced by the processor at its output pins. Such processors may be more difficult to debug, but the problem is still relatively manageable.

[0003] For processors that implement FRC, the reset state can have a significant impact on debug, verification and operation. One mechanism for implementing FRC replicates the instruction execution cores of a processor, runs the same instruction code on each processor core and compares results from the different cores at one or more stages of execution. If a discrepancy is detected, the computer system enters an error-handling mode, which may require reset of one or both pipelines. Current processing technologies allow the replicated cores to be implemented on a single processor die (multi-core processors).

[0004] The points of comparison between results from the different execution cores form the FRC boundary. The FRC boundary typically includes many more signals than appear on the processor's input/output (I/O) pins to provide reliable checking. The machine states of a processor that implements FRC are thus examined at a significantly finer level of detail than those of a non-FRC processor. Indeterminate states in the nodes of one or the other execution core can lead to unnecessary mismatches at the FRC boundary.

[0005] One strategy for addressing this problem is to run a reset code module in response to a reset event. The reset code steps the processor through a precise sequence of operations that force targeted processor nodes to specified logic states. More nodes may be driven to specified states by using ever larger and more complex reset codes, but developing, debugging and validating such reset codes is very costly.

[0006] The present invention address these and other problems associated with generating deterministic reset states for processors.

Brief Description of the Drawings

[0007] The present invention may be understood with reference to the following drawings, in which like elements are indicated by like numbers. These drawings are provided to illustrate selected embodiments of the present invention and are not intended to limit the scope of the invention.

[0008] Fig. 1 is a flowchart providing a high level view of a reset operation typical of a non-FRC processor.

[0009] Fig. 2 is a block diagram of one embodiment of a computer system suitable for implementing the present invention.

[0010] Fig. 3 is a block diagram representing one embodiment of the reset module shown in Fig. 2.

[0011] Figs. 4A and 4B are block diagrams showing embodiments of a reset module driving multiple scan chains in parallel and serial configurations, respectively.

[0012] Fig. 5 is a flowchart representing on embodiment of a method in accordance with the present invention for establishing the reset state of a processor.

Detailed Description of the Invention

[0013] The following discussion sets forth numerous specific details to provide a thorough understanding of the invention. However, those of ordinary skill in the art, having the benefit of this disclosure, will appreciate that the invention may be practiced without these specific details. In addition, various well-known methods, procedures, components, and circuits have not been described in detail in order to focus attention on the features of the present invention.

[0014] The machine state of a processor is determined by the states of its nodes. These nodes include the voltages on latches, flip-flops and like components of various storage and control logic of the processor. A reset mechanism places a processor in a known machine state before the processor is allowed to begin executing code freely. This specified starting point ensures that the processor's output is reliable and reproducible.

[0015] A non-FRC processor may operate reliably by driving a relatively small number of its many nodes to specified states. The unspecified nodes do not affect the results generated by the processor once it begins operating after reset. The nodes that are required to be in specified states may be handled through a hardware reset tree and a relatively modest reset code. The reset code may be stored in firmware accessible to the processor on reset.

[0016] Fig. 1 is a flowchart representing a reset operation for a non-FRC processor. A reset signal is detected 110 by, for example, asserting a voltage level on a processor pin (reset pin). The reset signal is propagated 120 down a reset tree to drive various nodes of the processor to specified states. Propagation of the reset signal typically reaches only a small fraction of the nodes of a processor. Even for a non-FRC processor, the number of nodes reached through a reset tree is generally not sufficient to ensure that the processor provides reliable results when it begins operations. For these processors, control transfers 130 to an entry point for reset code indicated by a RESET vector. The processor executes 140 the reset code to drive additional nodes to specified states. Some processors may employ resettable latches at all or most nodes that require specification on reset, which reduces or eliminates the reset code required.

[0017] As noted above, the minimum reset state necessary to provide reliable, reproducible results for a processor that implements FRC is significantly larger than that necessary for a non-FRC processor. The present invention provides a flexible mechanism for establishing a reset state in an FRC processor.

[0018] Fig. 2 is a block diagram of one embodiment of a computer system 200 in which the present invention may be implemented. Computer system 200 includes one or more multi-core processors 210, a chipset or system logic 270, main memory 280, a non-volatile memory 290, one or more peripheral devices 294. Chipset 270 controls data transfers among processor(s) 210, main memory 280, non-volatile memory 290, and peripheral devices 294. Computer system 200 is provided to illustrate various features of the present invention. The particular configuration shown is not necessary to implement the present invention.

[0019] The disclosed embodiment of processor 210 includes first and second execution cores 220(1) and 220(2), respectively, a reset module 240, an FRC checker 250, and a test access port

(TAP) 260. Each execution core 220 includes one or more scan chains 230 that may be accessed through a port 260. For one embodiment of system 200, port 260 may be a test access port (TAP) as specified by IEEE standard 1149.1 and its revisions. FRC checker 250 represents the FRC boundary for the disclosed embodiment of processor 210. As indicated in the figure, the FRC boundary is internal to the die on which processor 210 is formed, where it can receive signals reflecting the state of data from execution cores 220.

[0020] Reset module 240 controls scan chains 230 in response to a reset signal. As discussed below in greater detail, reset module 240 uses scan chains 230 to force various nodes of execution cores 220 that are accessible through scan chains 230 into specified states. Reset module 240 is shown as a single unit for convenience. Embodiments of system 200 may include a reset module 240 for each execution core 220 or for different subsets of execution cores 220 if processor 210 includes more than two execution cores 220.

[0021] Fig. 3 shows in greater detail one embodiment of reset module 240 and scan chain 230. Scan chain 230 includes processor nodes 350(1)-350(j) (generically, nodes 350) that are connected in series through a data line DATA and a clock line CLK. Processor nodes 350 represent latches, flip-flops, and comparable devices for storing data and signals that accompany instructions in process by execution core 220. Generally, the nodes included in a scan chain are selected from the execution, staging and control logic of processor 210 to provide control points that may be programmed and monitored for testing/debugging a processor. For example, a node or sequence of nodes may specify a state or data register for the processor.

[0022] For the present invention, scan chain 230 may include additional nodes 350 selected to provide a more deterministic reset state for processor 210. During testing/debug, scan chain 230 allows data and/or instructions to be loaded into selected nodes 350. Processor 210 is then

stepped through execution cycles and values in various nodes 350 may be captured for analysis. A typical scan chain may include thousands of such nodes and an execution core may have multiple scan chains. During RESET, scan chain 230 provides an additional function. Embodiments of processor 210 may include scan chains 230 that are used exclusively for debug/validation, exclusively for reset or for both debug/validation and reset.

[0023] The disclosed embodiment of reset module 240 includes a clock mux 310, a clock mode selector 320 and a RESET pattern generator 330. Clock mode selector 320 detects a reset condition and provides a signal to a control input of clock mux 310. Clock mux 310 provides a clock signal to the clock line (CLK) of scan chain 230, responsive to the signal from selector 320. For example, if processor 210 is in its normal operating mode, clock inputs (C) of processor nodes 350 are driven by a core clock signal (C_CLK). If a reset condition is triggered, nodes 350 are driven by a scan clock signal (S_CLK). Generally, S_CLK will have a lower frequency than C_CLK due to the physical limitations of scan chain circuitry. For the disclosed embodiment of reset module 240, mode selector 320 triggers clock mux 320 to provide S_CLK to nodes 350 in response to a reset signal, and C_CLK to nodes 350 if no reset signal is detected.

[0024] Reset pattern generator 330 provides a bit pattern to the data line (DATA) of scan chain 230, using S_CLK. The bit pattern is driven to nodes 350 of scan chain 230 to put them in a deterministic state. For example, reset pattern generator 330 may output a string of 1's, a string of 0's or a string including some pattern of 1's and 0's, e.g. 101010101….. For many nodes, the exact values driven matter less than the fact that as a result of reset module 240, each node 350 on scan chain 230 will be in a defined state. By applying the same bit pattern to both execution cores 220 of processor 210, the reset states of corresponding nodes 350 are identical. This reduces the potential for discrepancies between the reset machine states of execution cores 220,

making their combined behavior more predictable. The likelihood of FRC checker 250 detecting a result mismatch attributable to differences in the reset machine states of execution cores 220 is greatly reduced. The more extensively scan chain 230 cover nodes 350 of processor 210, the less opportunity there is for such discrepancies to arise.

[0025] Fig. 4A is a block diagram representing a reset system 400 in which reset module 240 drives a parallel configuration of scan chains 230(1)-230(n). Reset system 400 allows nodes 350 of scan chains 230(1) –230(n) to be placed in their reset states in parallel. For example, for dual core processor 210, n = 2, and corresponding nodes 350 in scan chains 230(1), 230(2) of execution cores 220(1), 220(2), respectively, may be driven to their reset states in parallel. Of course, separate reset modules 240 may be used for each execution core 220 and the same result will result provided pattern generators 330 are properly matched.

[0026] Fig. 4B is a block diagram representing a reset system 450 in which reset module 240 drives a series configuration of scan chains 230(1)-230(k). The disclosed configuration may be useful, for example, within a given execution core 220, since each execution core 220 may have multiple scan chains 230.

[0027] Hybrids of systems 400 and 450 may also be implemented, in which reset module 240 drives scan chains 230 in various combinations of parallel and serial configurations. For example, reset module 240 may drive corresponding scan chains 230 of each execution core 220 in parallel, while driving different scan chains 230 within each execution core in series. That is the first scan chain in execution cores 220(1)-120(n) may be driven in parallel, followed by the second scan chains in execution cores 220(1)-120(n) et seq.

[0028] Fig. 5 is a flow chart representing one embodiment of a method 500 for resetting the execution cores of a multi-core processor. A reset signal, triggered by, for example, a cold start

(cold reset) or an FRC mismatch (warm reset) is detected 510. Responsive to the reset signal, nodes on scan chains of each execution core are driven 520 to specified states. For example, a reset module applies a scan clock to the nodes of the scan chain and an associated pattern generator drives a bit pattern onto the nodes. For an FRC processor, the same bit pattern drives corresponding scan chains in each execution core. Typically, a reset signal is propagated 530 to any nodes on a hardware reset tree to drive them to specified states. Further, reset code, if present, is executed 540 to drive additional nodes to their reset states.

[0029] For one embodiment of method 500, the source of the reset event is checked 550. If the reset event is an "external reset" such as a cold reset or other platform generated reset, a reset algorithm may be executed and a rendezvous state entered 560. The reset algorithm may include, for example, tests of the processor, system components, and the like. If the reset is an internal reset, e.g., in response to an FRC mismatch, method 500 may execute a correction routine 570 to address the source of the mismatch and return 580 to execution at the point at which it was interrupted.

[0030] For one embodiment of computer system 200, processor(s) 210 may be configured in a high reliability mode or a high performance mode. The mode may be selected when system 200 is booted. Some embodiments of system 200 may be able to convert between modes on the fly, e.g. after boot. For the high reliability mode, FRC checker 250 is activated and execution cores 220 operate on identical code streams to implement FRC. For the high performance mode, FRC checker 250 is deactivated, and execution cores 220 operate on different code streams, effectively doubling the throughput of processor 210. For this embodiment of computer system 200, reset mechanism 100 may be implemented on reset if the processor is in high performance

mode, and reset mechanism 500 may be implemented on reset if the processor is in high reliability, i.e. FRC, mode.

[0031] There has thus been provided a mechanism for establishing the reset state of a processor, using scan chains. In response to a reset event, a reset module drives specified bit patterns onto the scan chains to set the states of the component nodes (latches, flip-flops, and the like). For multiple execution cores in FRC mode, identical patterns on driven onto corresponding scan chains in each execution core. The resulting reduction in unspecified nodes provides greater control over the reset machine state of the processor. For an FRC processor, this translates to a lower likelihood of FRC mismatches attributable to indeterminate node states.

[0032] The disclosed embodiments have been provided to illustrate various features of the present invention. Persons skilled in the art of processor design, having the benefit of this disclosure, will recognize variations and modifications of the disclosed embodiments, which none the less fall within the spirit and scope of the appended claims.